An Efficient and Responsive Robot Motion Controller for Safe Human-Robot Collaboration

Xuan Zhao[1,4], Tingxiang Fan[2,4], Yanwen Li[3], Yu Zheng[4†], Jia Pan[2†]

***Abstract*—Safety and efficiency are two crucial factors for human-robot collaboration. It is challenging to ensure human safety while not sacrificing the task efficiency. In this paper, we present a reinforcement learning (RL) based method with a hazard estimator to balance these two factors. Our method has two phases. In the training phase, an RL control policy and a hazard estimator are trained; in the testing phase, we dynamically select a guiding goal along a given task path to balance between human avoidance and task execution. The proposed method is compared among three previous methods: another RL based method, a reactive method, and a motion planner both in simulated and real-world experiments. Results show that our method can 1) enable a robot to follow a demonstrated (reference) path if the human stays far from the robot; 2) apply responsive online motion adaption to balance human avoidance and task efficiency if the human moves closer toward the robot. In addition, the dynamic goal selection method is easy to use, and can effectively increase the success rate and provide a better trade-off between safety and efficiency.**

***Index Terms*—Human-Robot Collaboration, Human-Aware Motion Planning, Safety in HRI**

I. Introduction

MANY manufacturing tasks such as assembly require a high level of dexterity and flexibility beyond the capability of the state-of-the-art autonomous robotics technique, and thus a human worker's involvement is indispensable. To guarantee safe collaboration with humans, most of the state-of-the-art collaborative robots operate at a safe speed and must be turned into the emergency stop mode once a collision is detected. Such stops could significantly interrupt the workflow of both robots and human co-workers. To avoid this issue, robots must actively avoid humans, which is challenging because the human's existence makes the workspace a scenario with dynamic obstacles.

There are several solutions allowing robots to operate in dynamic environments. One is to perform frequent replanning according to sensory feedback [1], [2]. However, replanning is computationally expensive with significant delays. Limited by the time budget, the online replanner may only find a sub-optimal solution in the case where the robot arm may collide with the human. The replanning solution can be improved if a high-quality human motion prediction is available [3], though it is difficult in practice due to the complexity of human behaviors. Another popular approach for human-robot collision avoidance is to use a distance/potential field based collision risk [4] to generate a safe reactive policy of the robot that responds quickly to scenario changes [5], [6]. But as a greedy approach, the generated collision avoidance policy may significantly reduce the task efficiency.

Fig. 1: The real-world assembly task. Left: a robot controller without human awareness, and the robot interrupts human's work. Right: our human-aware robot controller and the robot leaves human workspace clear while moving towards the target.

Machine learning provides an alternative to handle the limitations of previous methods. More specifically, reinforcement learning has been explored for robot motion adaption and motion planning. For instance, [7] generates a smooth robot trajectory according to a reference path. [8] and [9] generate robot motions by taking into account environmental constraints, such as behaviors of other agents or human preference. Moreover, [10] shows the potential to control a robot via deep reinforcement learning (DRL) in a dynamic environment, though it is limited to small robot workspace and repetitive obstacle motions. In this paper, we avoid these limitations by using a training environment with human-alike agents and a hazard aware goal selection, to make DRL better adapt to the human-robot collaborative (HRC) tasks.

In this paper, we propose a DRL based method that could 1) enable a robot to follow a demonstrated (reference) path if the human stays far from the robot; 2) apply online motion adaption to balance human avoidance and task efficiency if the human stays closer to the robot. We take a reference path as the input rather than planning directly by RL because in most HRC scenarios, a path can be computed by many existing methods with lower computational cost than that of RL. For example, most collaborative robots have the function to record demonstrations; or an offline motion planner can generate reference paths that respect human preferences.

Manuscript received: January, 11, 2021; Revised April, 12, 2021; Accepted May, 26, 2021.

This paper was recommended for publication by Editor Gentiane Venture upon evaluation of the Associate Editor and Reviewers' comments. This work was partially supported by General Research Fund (GRF) 11207818, 11202119 and NSFC-RGC joint grant N_HKU103/16.

[1] Department of Biomedical Engineering, City University of Hong Kong

[2] Department of Computer Science, The University of Hong Kong

[3] Department of Systems Engineering and Engineering Management, City University of Hong Kong

[4] Tencent Robotics X, Shenzhen, China

† denotes the corresponding author. (Jia Pan: jpan@cs.hku.hk, Yu Zheng: petezheng@tencent.com)

Digital Object Identifier (DOI): see top of this page.

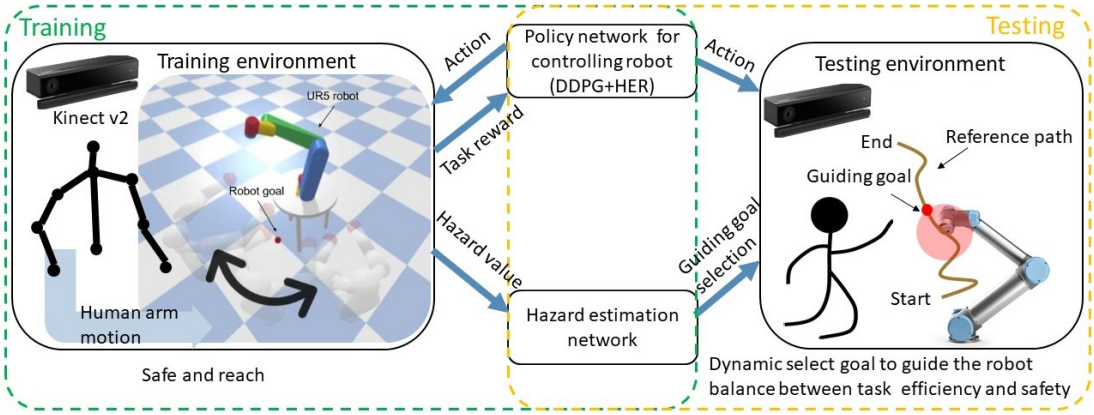

Fig. 2: The framework of our system. In the training phase, the robot learns to reach a static goal and avoid humans; in the testing phase, the robot dynamically selects a guidance goal along the reference path for switching between path following and human avoidance.

In addition, since the robot task is always repetitive in an industrial environment, the robot motion can be more consistent if it always moves close to a reference path. Thus, our method focuses on the online motion adaption between human safety and path following. Our method consists of two phases: offline training and online testing. In the training phase, the robot learns to reach a static goal and avoid humans; in the testing phase, we dynamically select a guidance goal along the reference path to guide the robot switching between following the path and avoiding humans. We test and compare our method with three baseline approaches in both simulation and real-world experiments, including a previous RL method, an optimization based motion planner, and a distance based controller. Results show that our method has a higher success rate than previous RL methods. It also responds faster than the optimization-based methods and achieves a better task efficiency than the reactive controller. An overview of our method is shown in Figure 2.

This paper is organized as follows. Section II introduces the background of RL used in this paper. Section III shows the environment setting for training. The proposed method is detailed in Section IV. Experiment results and conclusions are given in Section V and VI respectively.

II. Background

We consider the standard RL setting that consists of an agent interacting with a stochastic environment E in discrete timesteps. The environment is modeled as a Markov decision process with a state space $\mathcal{S}$, a set of actions $\mathcal{A}$, an initial state distribution $p(s_1)$, a transition probability $p(s_{t+1} \mid s_t, a_t)$ and a reward function $r(s_t, a_t)$. At each time step t, the agent receives an observation of state s_t, takes an action a_t, and receives a scalar reward $r_t \in \mathbb{R}$.

An agent's behavior is defined by a policy $\pi(a_t \mid s_t)$, which is a mapping from observations to actions: $\pi : \mathcal{S} \rightarrow \mathcal{A}$. The return from a state is defined as the sum of discounted future reward $R_t = \sum_{i=t}^{T} \gamma^{(i-t)} r(s_i, a_i)$ with a discounting factor $\gamma \in [0, 1]$. The goal in RL is to learn a policy which maximizes the the expected return from the start distribution $J = \mathbb{E}_{r_i, s_i \sim E, a_i \sim \pi}[R_1]$.

DDPG [11] is a model-free Q-learning-based RL algorithm for continuous action spaces. In DDPG, we maintain two neutral networks: a deterministic policy (called the actor) π and a Q function approximator (called the critic), parameterized by a set of parameters θ^{π} and θ^{Q}. An actor network deterministically maps observations to actions and tries to maximize $Q(s_t, a_t)$. Each transition of the agent is stored in a replay buffer R, from which mini-batches are sampled to train the networks. In order to improve the sample efficiency from sparse reward, Hindsight Experience Replay (HER) [12] suggests an additional replay technique. In this technique, besides standard experience replay, it also samples a set of additional goals for replay $G := \mathbb{S}(\text{current episode})$. For each $g' \in G$, it calculates $r' := r(s_t, a_t, g')$ and stores this new transition $(s_t \parallel g', a_t, r', s_{t+1} \parallel g')$ in R.

III. Environment Setting

To achieve good performance using DRL, the training environment should be able to well encode the target human-robot collaborative task and reflect the safety requirements during the collaboration. Previous work [10] uses three obstacles moving from random start positions toward the robot goal position to pose suitable constraints for human-robot collaboration, but it is not ideal since the human motion is constrained by the skeleton and thus difficult to well be approximated by the motion of a set of random obstacles. Such approximation also leads to an overly conservative robot movement. In our system, we directly record real skeleton data of human's upper body by a Kinect v2 camera and map these data to a humanoid model which is then added in the simulated environment to pose human-robot collaboration constraints related to both task and safety. Here the data recording does not involve the robot motion because our goal is to make the robot actively adapt to and minimally interrupt the human's motion so that the human can make decisions as if the robot were not existing. We designed two tasks for this recording: 1) a toy assembly task, and 2) random arm moving and waving for better generality. We invited three participants for the recording. For each of them, we recorded 4-minute data for the assembly task and 1-minute for the random movement task. Data from two participants is used as the training data, and the other is used as the validation data. Note that here we assume the human and robot are collaborating in a industrial scenario where the human's behavior is strongly restricted by the task

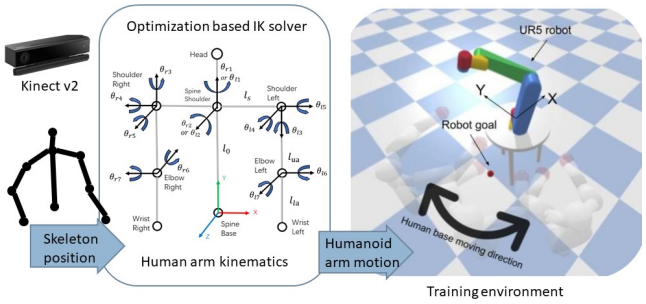

Fig. 3: Our training environment includes a UR5 robot, the workplace with a table, and a humanoid obstacle. The robot's initial and goal poses are sampled randomly with the its reachable region. The motion of the humanoid obstacle includes a circular locomotion and a recorded arm motion.

and environment. In this way, the robot's optimal responsive policy is also restricted and thus is learnable using a small data set of three participants. This is for sure not sufficient for learning a general human-robot collaboration behavior, which is out of the scope of this paper.

A. Human motion in environment

For the skeleton data collected from the camera, we can only use its joint position values because the joint orientation is quite noisy. To map the skeleton position values to a humanoid agent in simulation, we solve an optimization that minimizes the joint position error between the real human and the humanoid. Because we are more interested in the relative movement of human's two arms, we omit the rotation of the human spine to simplify the calculation. In particular, we use a simplified version of the humanoid kinematics as shown in Figure 3, where we assume the coordinate of spine base coincides with the same joint of the real human, and each arm consists of 7 revolute joints (2 for spine shoulder, 3 for shoulder and 2 for elbow) and 1 end position (wrist).

To minimize the joint position error, we first set the spine base frame as the humanoid base and calculate the analytical forward kinematics for each joint. Then we use Quasi-Newton to find the optimal joint angles that minimize the error between recorded skeleton position and the position calculated using forward kinematics. Finally, we map these joint angles to the humanoid model in simulation to obtain human-like motions.

B. Human-robot setting in simulation

The training environment includes a 6 DoF UR5 robotic arm, a workspace with a table, and a humanoid skeleton, as shown in Figure 3. The initial and goal positions of the robot's end-effector are sampled randomly within the robot's reachable region, with the orientation fixed. The initial robot joint configuration is calculated via inverse kinematics.

To encourage the robot to efficiently learn a safe movement policy, the training environment should be hazardous for the robot, i.e. the human motion should be sufficiently complicated so that it is nontrivial for the robot to learn a high quality collision avoidance policy. For this purpose, we design the human motion with both locomotion and arm motion. The arm motion is generated in the way as described in Section III-A. The locomotion is created by moving the humanoid's base in a circle around the table at a speed up to $0.5\,\mathrm{m\,s^{-1}}$. To generalize among people with different sizes, the humanoid's height is randomly set between $1.3\,\mathrm{m}$ and $1.8\,\mathrm{m}$.

C. Observation

The DRL state s_t consists of the robot state o_t^R, the human state o_t^H, and the task state o_t^T. The robot state o_t^R includes the robot's current end-effector position $\boldsymbol{p}_t^{ee}$, current joint angles $\boldsymbol{q}_t^r$, and joint angular velocity at the last 2 time steps $[\dot{\boldsymbol{q}}_{t-1}^r, \dot{\boldsymbol{q}}_t^r]$. The task state o_t^T includes the end-effector position and joint angle of robot goal $\{\boldsymbol{p}^g, \boldsymbol{q}^g\}$. The human state o_t^H includes the distance vectors between positions human joints and robot end-effector at the last 10 time steps. In addition, when the human stands far away from the robot, the distance vector can be very large, which increases the difficulty of training the neural network. To avoid this, we clip the maximum absolute distance to 1 and calculate the human status observation as $o_t^H = \left\{\boldsymbol{d}_\tau^{hj}\right\}_{hj\in HJ, \tau\in t-10:t}$ as,

$$\boldsymbol{d}_\tau^{hj} = clip[(\boldsymbol{p}_\tau^{hj} - \boldsymbol{p}_\tau^{ee}), -1, 1] \tag{1}$$

where $\boldsymbol{p}_\tau^{hj}$ denotes Cartesion position of human's joint hj at time step τ, and $hj \in HJ$ ={left shoulder, left elbow, left hand, right shoulder, right elbow, right hand}. In order to measure position of human's joints, a Kinect v2 camera is installed at a fixed location facing the human. Then the measurements are transformed to the world frame from camera frame. As a result, $o_t = [o_t^H, o_t^R, o_t^T]$ denotes the observation of the current state of the environment. The dimensions of o_t^H, o_t^R, and o_t^T are $180 = 18 * 10$, 21, and 9, respectively.

D. Action

The action a_t consists of the joint angular velocity of the robot, because unlike [10] which controls the velocity of the end-effector, we have a full 6 DoF control of the robot. This also saves us from the trouble of mapping the end-effector velocity to the joint velocity.

E. Reward

We consider the standard RL problem described in Section II. To get a safe reach motion, we include both the reach and collision information into a reward function. As pointed out in previous work [13], a close proxemic distance will reduce the human efficiency. Thus, we include a distance term in the reward to push the robot away from the human. Besides, a smoothness term is added to the reward to make a smooth control. The reward function is formulated as:

$$\begin{aligned} r =& \alpha_1 \cdot [d(x_{target}, x_{robot}) > \delta] + \alpha_2 \cdot [collision = 1] \\ &+ \alpha_3 \cdot c_{dist} + \alpha_4 \cdot c_{smooth}, \end{aligned} \tag{2}$$

where α_i is important weight of different terms. $d(\cdot)$ measures the Euclidean distance between $x_{target} = \boldsymbol{q}^g$ and $x_{robot} = \boldsymbol{q}_t^r$. The task is considered as successful is $d(x_{target}, x_{robot}$ is smaller than a threshold δ. $collision$ checks whether the robot is hitting obstacles or in a self-collision pose. c_{dist} is the cost for proxemic distance between human and robot, and c_{smooth}

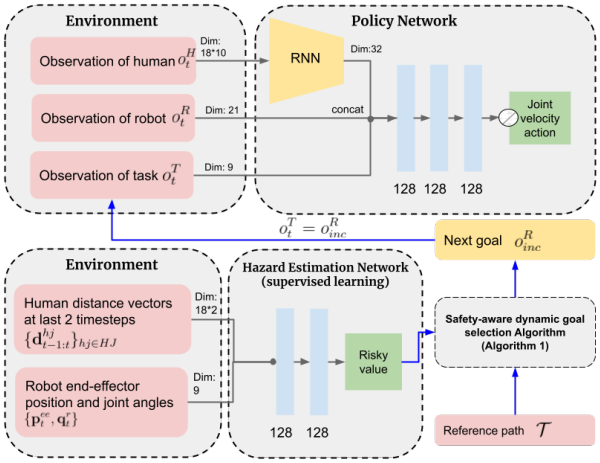

Fig. 4: Network architecture. Blue line indicates the additional goal selection process in the testing phase. Upper: The policy network, including a single layer RNN with 32 GRU unit for extracting human temporal features and three fully connected layers. The activation function for the output layer is linear. Lower: The hazard estimation network, which has two fully connected layers, and the activation function for the output is also linear.

is the cost for smoothing robot action. These two costs are defined as:

$$c_{dist} = \begin{cases} d^{safe} - \min(\|\boldsymbol{d}_t^{hj}\|), & \|\boldsymbol{d}_\tau^{hj}\| \leq d^{safe}, \\ 0, & \text{otherwise}, \end{cases} \quad (3)$$
$$c_{smooth} = \|\dot{\boldsymbol{q}}_{t-1}^r - \dot{\boldsymbol{q}}_t^r\|,$$

where d_{safe} is a manually chosen safe distance for human and in our experiments we set $d_{safe} = 0.15$.

IV. Approach

A. Training

The training phase consists of a DDPG+HER algorithm for safe robot motion training and a supervised learning algorithm for training collision checking network. The corresponding dual-network architecture is shown in Figure 4. The policy network includes a single layer RNN with 32 GRU unit to better extract human temporal features as well as three fully connected layers. The output of RNN is concatenated with the observation of the robot and the task. The concatenated data then is processed through the fully connected layers, and the final output is the robot joint velocity. As described in Section II, the goal of the DDPG algorithm is to learn a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ that maximizes the expected return. It is worthy noting that reinforcement learning learns a long horizon policy, so the agent can learn a global replanning behaviour rather than a simple reactive action.

Since we want the robot to balance path following and human avoidance, we also train a hazard estimation network to online evaluate the risk of hitting the human. The hazardous value consists of two parts: the collision value and the safe distance value, and both parts are part of the reward function designed in (2). Thus, the hazardous value is formulated as $v^{risk} = \alpha_2 \cdot [collision = 1] + \alpha_3 \cdot c_{dist}$. To improve data efficiency, we save these two reward component separately in RL replay buffer and reuse the buffer's data to train the hazard estimation network. The hazard network's input includes: the distance vectors of human at the last 2 timesteps $\{\boldsymbol{d}_{t-1:t}^{hj}\}_{hj \in HJ}$, robot's current end-effector position and joint angles $\{\boldsymbol{p}_t^{ee}, \boldsymbol{q}_t^r\}$. The neural network is designed to minimize the loss: $L = \sum_i \left\| y_i - v_i^{risk} \right\|$, where i is the index of the data in a batch.

There are two reasons we use a hazard estimation network instead of a traditional collision checking algorithm. First, it is time-efficient because the risky value can be calculated directly from the raw data without updating the human state in a simulator. Second, the terms "reach" and "safe" are coupled together in the reward formulation. If we set a high weight on "safe", the policy can be too lazy and the robot can not reach the goal. If we set more weight on "reach", the robot can reach the goal but its behaviour can be very aggressive when it is close to the goal. Thus, it is hard to balance between "reach" and "safe". The introduction of the hazard network decouples the risky reward component from the entire reward, which is helpful for selecting a good guiding goal, as will be discussed below.

B. Dynamically selecting goals from a demonstrated path

To guide the robot to balance between following demonstration path and bypassing the obstacles, we suggest a dynamic safety-aware goal selection algorithm in each control loop, shown in Algorithm 1. Firstly, we use Algorithm 2 to find the intersection on the reference path whose distance to the robot's current end-effector position $\boldsymbol{p}_t^{ee}$ is r. We set $\Delta r = 0.2$ and $r_{large} = 0.5$. To evaluate the risk for the robot moving to the intersection point, we linearly interpolate 10 observations between the current robot state and the intersection. Then we calculate the risk value v_i^{risk} of all observations by querying the hazard estimation network in parallel and add their absolute values together. The risk value includes the collision cost and distance cost so that its absolute value is large if the robot has a high tendency to hit the human. We manually choose 0.5 as the risk threshold. If the risk value is higher than the threshold, we set a large radius and choose the robot goal far from the current robot position. If the risk is lower than the threshold, the robot can move along the given path, so the robot goal is set close to the current robot configuration. A more intuitive explanation of the goal selection procedure is shown in Figure 5. The dynamic goal selection has benefits in threefold:

1) The robot can strictly follow a demonstrated path if no human is in close proximity.
2) If human blocks the demonstrated path, the robot will bypass the human and then continue its work. The dynamic goal selection helps the policy choose a direction that could quickly finish the task while avoiding the human.
3) The "reach" and "safe" reward terms are coupled in the reward formulation, and thus the robot would be more aggressive if it moves closer to a target. With the dynamic goal selection, a guidance goal would be set further if the evaluated risk is high, and thus the chances of aggressive behaviour will be reduced.

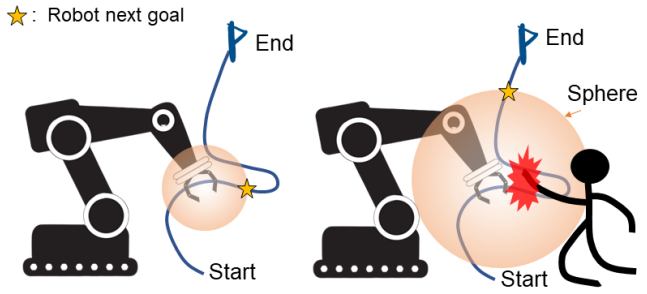

Fig. 5: An intuitive representation of robot dynamic goal selection

Algorithm 1 Select a safety-aware dynamic goal

Input:
- Current robot status observation $\boldsymbol{o}_t^R$.
- Current human status observation $\boldsymbol{o}_t^H$
- Hazard estimation network $\mathbf{H}$.
- Intersection function $\mathbf{F}$ depicted in Algorithm 2
- Reference path $\mathcal{T} = \{\boldsymbol{o}_n^R\}_{n=1:N}$ with N waypoints.

// set an initial searching radius
$r = 0.1$
// find intersection
$\boldsymbol{o}_{inc}^R = \mathbf{F}(\mathcal{T}, \boldsymbol{o}_t^R, r)$
while not find intersection **do**
  // increase the searching radius
  $r = r + \Delta r$.
  $\boldsymbol{o}_{inc}^R = \mathbf{F}(\mathcal{T}, \boldsymbol{o}_t^R, r)$
end while
// interpolate 10 robot states between the intersection and the robot state
$\{\boldsymbol{o}_i^R\}_{i=1:10} = Interpolate(\boldsymbol{o}_{inc}^R, \boldsymbol{o}_t^R)$.
// get batch risk values
$\{v_i^{risk}\}_{i=1:10} = \{\mathbf{H}(\boldsymbol{o}_t^H, \boldsymbol{o}_i^R)\}_{i=1:10}$
// if the risk is high, a larger radius is set
if $\sum_{i=1}^{10} \left|v_i^{risk}\right| > 0.5$ and $r < r_{large}$ **then**
  $r = r_{large}$.
  $\boldsymbol{o}_{inc}^R = \mathbf{F}(\mathcal{T}, \boldsymbol{o}_t^R, r)$.
end if
Return $\boldsymbol{o}_{inc}^R$

Algorithm 2 Calculate intersection

Input: reference path $\mathcal{T}$, robot current state $\boldsymbol{o}_t^R$ contains $\boldsymbol{p}_t^{ee}$, sphere radius r.
Find waypoint ids M that satisfy: $\|\boldsymbol{p}_n^{ee} - \boldsymbol{p}_t^{ee}\| \leq r$.
if $M \neq \emptyset$ **then**
  Find the last id m from the first consecutive integer sequence of M.
  Calculate Cartesian interactions $\boldsymbol{p}_{inc}^{ee}$ between straight line with point $\{\boldsymbol{p}_m^{ee}, \boldsymbol{p}_{m+1}^{ee}\}$ and sphere centered at $\boldsymbol{p}_t^{ee}$ with radius r.
  Linear interpolate on joint waypoints to get joint intersection $\boldsymbol{q}_{inc}^r$according to $\boldsymbol{p}_{inc}^{ee}$.
  Return $\boldsymbol{o}_{inc}^R = \{\boldsymbol{p}_{inc}^{ee}, \boldsymbol{q}_{inc}^r\}$.
else
  Return none.
end if

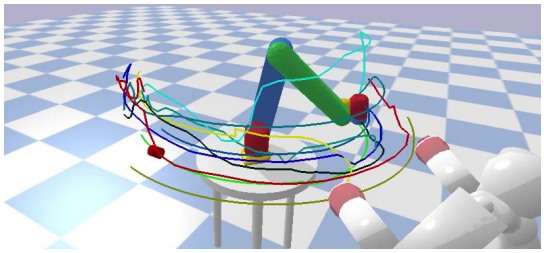

Fig. 6: Reference paths for testing.

V. Experiment and Results

Our experiments include three parts: the training results of the proposed two networks, the quantitative comparison in simulation among three previous approaches and our method, and finally, a set of real-world experiments demonstrating the performance of our proposed method. All the methods was implemented in Pybullet [14] simulator. The three baseline approaches considered in our experiments are:

1) An RL based method for safe human-robot interaction [10].
2) A distance based reactive method [15]. We re-implement the "safety control module" of this method in our environment. In this work, the robot end-effector's velocity v_{ee}^r is calculated as:

$$\dot{\boldsymbol{p}}^{ee} = k_g(\boldsymbol{p}^T - \boldsymbol{p}^{ee}) - \sum_{hj \in HJ} \left(f_D^{hj} * f_V^{hj} * (\boldsymbol{p}^{ee} - \boldsymbol{p}^{hj}) \right),$$

where $\boldsymbol{p}^g$, $\boldsymbol{p}^{ee}$ and $\boldsymbol{p}^{hj}$ are positions of the goal, the robot end-effector and the human joints, respectively, and $k_g = 5.0$. f_D and f_V are calculated as described in [15]. In addition, if $\|\dot{\boldsymbol{p}}^{ee}\| > \dot{p}_{\max}^{ee}, \dot{\boldsymbol{p}}^{ee} = \dot{p}_{\max}^{ee} \cdot \dot{\boldsymbol{p}}^{ee} / \|\dot{\boldsymbol{p}}^{ee}\|$, where $\dot{p}_{\max}^{ee} = 1.0$.
3) A replanning method, more specifically, an optimization based motion planner ITOMP [2]. The original ITOMP paper does not specify which optimization solver shall be used. Since the cost of dynamic obstacles in our formulation is non-differentiable, we use STOMP [16] as the solver.

We record 10 reference paths using the UR5 teaching mode and randomly select one of them for each trial, shown in figure 6. Every method has been tested for 500 trials. To reach a fair comparison, the human motions should be the same in tests for different methods. Because it is hard for a real human to repeat motions exactly the same, we use the recorded validation data for testing. Four metrics for quantitative comparison are designed as follows:

1) **Solving time**: the time spent to find the next action (for control based methods) or trajectory (for planning methods).
2) **Validation success rate**: the robot reaching target without collision on the validating set.
3) **Reaching time**: duration to reach the target.
4) **Trajectory length**: the distance swept by the robot's end-effector from start to goal.
5) **Joint tracking error**: Euclidean distance between the robot's joint angle to the reference path.

6) **End-effector tracking error**: Euclidean distance between the robot's end-effector position to the reference path.

All training and testing are run on an Intel(R) Core i7-7700 CPU with GeForce RTX 2060 GPU. For the hyperparametersin in Equation (2), We heuristically tried some numbers and selected the best combination from what we try. Finally we set $\alpha_1 = -1.5$, $\alpha_2 = -14$, $\alpha_3 = -6$, and $\alpha_4 = -0.5$.

A. Training result

Left side of Figure 7 shows the success rate over the number of training episodes. In our training environment, the robot always needs to avoid the human because the humanoid keeps moving even when the robot already reaches the goal. Thus, the success rate of an episode is calculated as: success rate $= \frac{\text{\#reach steps} - \text{\#collision steps}}{\text{\#total steps}}$, where each step takes $1/60\,\text{s}$ in the simulation. The #reach steps indicates the number of steps that the robot reaches the goal, and #collision steps indicates the steps that the robot is in collision. #total steps indicates the whole steps for one episode. It can be seen that the policy with GRU-MLP network converges faster and to a higher success rate than the MLP-only network. Right side of Figure 7 shows the training loss in terms of the collision value over the policy training episodes. The collision network is updated together with the policy network. Results show that it converges fast and successfully learns how to predict the collision value.

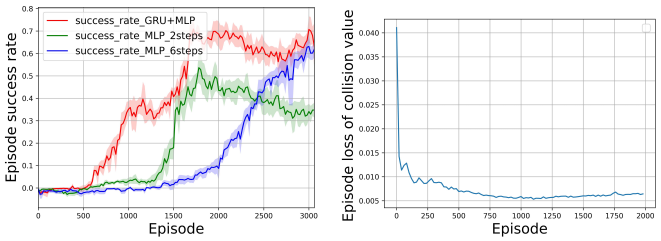

Fig. 7: Left: Curves of the average success rate of GRU-MLP and MLP-only network during training. 2steps and 6 steps indicate different input steps of human observation. Right: Curves of the hazard evaluation network's training loss.

B. Qualitative result

Figure 8 shows an example of the robot motion controlled by our human-aware motion controller. The yellow line is a reference path collected using the robot teaching mode, and the small red cylinder represents the robot's current goal. In the beginning, the risk of following the path is low, and thus the robot selects a guidance goal close to its current state. Later, the hazard network predicts that it is risky to strictly follow the path (as shown in the middle two figures). Thus, to bypass the human, the robot's guidance goal is set farther from the robot's current state. Finally, the human moves away from the robot and then the robot switches back to the path following strategy.

C. Quantitative result

Table I illustrates the comparison result on the four metrics. We can observe that our method has the highest success rate and lowest tracking error on the validation set among all methods. To highlight the importance of our hazard aware goal selection, we also integrate this dynamic goal selection with the previous RL method and test the performance. The result shows that the dynamic goal selection significantly improves the validation success rate. We also observe that the dynamic goal selection also leads to longer reaching time, which is because the avoidance behaviour would spend more time. The previous RL method did not consider human kinematics in the training environment, so its performance is limited in our case with humanoid obstacles. In addition, the previous RL method controls the linear velocity of the robot's end-effector, so it lacks of control of three rotational DoFs, which results in its higher joint tracking error comparing to our approach, even after the use of dynamic goal selection.

We also observe that the reactive controller has the lowest solving time while ITOMP has the longest solving time. It is worthy noting that, when integrating into our PyBullet-based simulator, we re-implement in python the ITOMP by replacing its solver with STOMP and its solving time for one iteration is slower (about $80\,\text{ms}$) than the reported result (about $17\,\text{ms}$) in the original ITOMP. Thus, here we only reported the average number of iterations to find a valid solution, because for which our re-implemented version is the same as the original ITMOP implementation. In addition, to adapt to our slower re-implemention of ITOMP, we slow down the humanoid's moving speed by 5 times using linear interpolation for the ITOMP test.

The reactive controller shows excellent behaviour on avoiding human, but it does not support the bypass behaviour, thus the robot needs to wait until the human leaves. In our validation set, most trials of the reactive controller have neither collision nor arrival in the limited steps. As a result, we did not count the success rate for the reactive method. The ITOMP method works well on replanning with the shortest trajectory length and a fast reaching time. However, due to the delay on both planning and collision checking, the human may collide with the robot when the computation is ongoing.

We further plot in Figure 9 the tracking error curves for four methods in Table I (except the reactive controller which cannot reach the goal), in both the scenarios with and without a humanoid obstacle. We can observe that the previous RL-based method has a large tracking error of joint angles even when no human is in presence, because 3 DoFs are lost when only controlling the end-effector's position. When the human moves closer to the robot, we can observe that the robot quickly responds to the human by temporarily deviating from the path to avoid collision, until the human moves away.

Overall, our experimental results show that the RL based methods respond slightly slower than the reactive method but significantly faster than the re-planning method. Besides, our method demonstrates the highest success rate, and we can conclude that it achieves a good balance between avoiding human and following demonstrations.

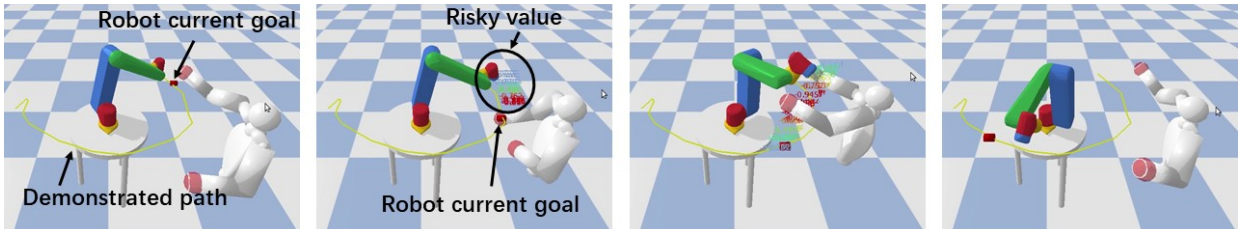

Fig. 8: Snapshots of the robot motion controlled by our human-aware motion controller. The robot states for the hazard evaluation is selected according to line 4 in Algorithm 1, and the hazard value is evaluated via the hazard evaluation network. For the risky value, the red color indicates a high risk while the green one indicates a low risk.

TABLE I: Comparison with three existing methods: Previous RL based method [10] , Distance based reactive controller [15], and ITOMP with STOMP optimizer [2]

	Validation success rate	Solving time (CPU)	Reaching time (s)	Trajectory length (m)	Joint tracking error	End-effector tracking error (m)
RL controller+dynamic goal (our method)	**86.4%**	4.8 ± 0.5ms	7.44 ± 1.36	1.94 ± 0.38	$\mathbf{0.252 \pm 0.143}$	$\mathbf{0.065 \pm 0.040}$
Previous RL based method [10]	60.8%	1.3 ± 0.2ms	$\mathbf{6.70 \pm 1.23}$	$\mathbf{1.45 \pm 0.25}$	1.522 ± 0.324	0.269 ± 0.067
Previous RL based method [10] +dymanic goal	75.4%	3.4 ± 0.3ms	7.82 ± 1.62	1.92 ± 0.33	1.213 ± 0.381	0.102 ± 0.064
Distance based reactive controller [15]	Can not reach if human keeps blocking the robot's way	$\mathbf{0.9 \pm 0.3}$ms	NaN	NaN	NaN	NaN
ITOMP with STOMP optimizer [2]	72.0%	47.6 ± 27.2ms	7.94 ± 2.8	1.98 ± 0.42	0.284 ± 0.138	0.073 ± 0.050

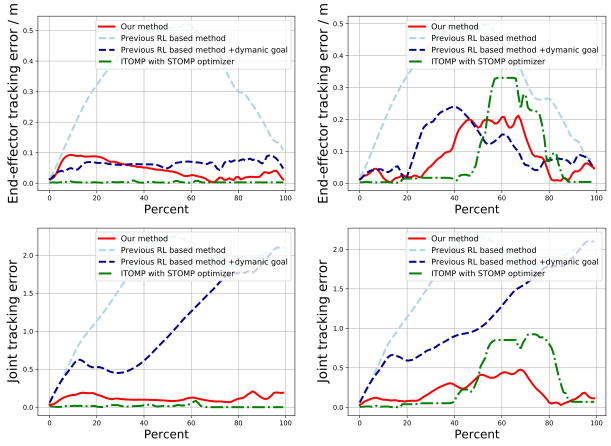

Fig. 9: Tracking error of robot motion w.r.t the demonstrated path in Figure 8. X-axis indicates the completion/goal-reaching percentage. Left: no human in presence; Right: robot with human in close proximity; Upper: the end-effector tracking error; Lower: the joint tracking error.

D. Ablation study

We also conduct an ablation study to evaluate the importance of each module proposed in this paper, and the result is shown in Table II. We use different color to depict the ordering of results from dark blue (best performance) to light blue (second-best performance).

1) GRU+MLP Policy + final goal: this model uses the GRU+MLP policy network without the guiding goal. The final goal is simply selected from the last pose of a given path;
2) GRU+MLP policy + static guiding goal: the guiding goal is selected by a fixed radius without any hazard evaluation;
3) GRU+MLP policy + dynamic guiding goal: the guiding goal is selected based on hazard evaluation;
4) MLP policy + dynamic guiding goal: this model uses the MLP policy with dynamic guiding goal selection. We choose two different input steps of human observation, i.e., 6 and 2.

Results show that adding a temporal encoder plays a significant role in improving the success rate and reaching time. The static goal selection causes a longer reaching time because the robot lacks knowledge of the entire path and therefore does not know the direction of either finishing tasks or avoiding human. The introduction of a hazard aware dynamic goal selection module can increase the success rate while only slightly reducing the tracking accuracy. This confirms the effectiveness of our method on balancing task efficiency and human safety.

E. Real-world experiment

We tested our methods in two real-world experiments, where the human's motion is captured by a Kinect v2 installed in front of the human. The first is a comparison among the reactive controller, the ITOMP replanning, and our method on a real UR5 robot with same reference path in a fixed time range. The second one is a cloth folding task where a human is asked to fold a cloth and the robot is moving in the human's proximity. Since collision checking is not necessary for reactive and RL methods, we don't need to map the human states to the humanoid agent in simulation for the real-world experiments. But such time-consuming mapping has to be performed for the ITOMP replanning for the collision checking.

The comparison result for the first experiment is shown in Figure 10. We can observe that using our method and ITOMP replanning, the robot can successfully bypass the human. Our method is advantageous in requiring less computational time and responding faster than replanning. For the reactive method, the direction of repulsive velocity from the human is opposite to the goal's attractive velocity. Thus, the robot moves away from the human, but its workflow is blocked unless the human moves away. The results for the second experiment is shown in Figure 1, where we compare a controller without human awareness, such as a strictly path following controller, and our method. Our method successfully leaves the human

TABLE II: Results of ablation experiments on the validation set

RL controller	Validation success rate	Reaching time (s)	Trajectory length (m)	Joint tracking error	End-effector tracking error (m)
GRU+MLP policy + final goal	79.4%	7.66 ± 2.40	1.96 ± 0.51	0.728 ± 0.328	0.166 ± 0.080
GRU+MLP policy + static guiding goal	81.4%	7.61 ± 1.40	$\mathbf{1.90 \pm 0.34}$	$\mathbf{0.190 \pm 0.160}$	$\mathbf{0.047 \pm 0.027}$
GRU+MLP policy + dynamic guiding goal	**86.4%**	$\mathbf{7.44 \pm 1.36}$	1.94 ± 0.38	0.252 ± 0.143	0.065 ± 0.040
MLP policy (6 steps) + dynamic guiding goal	70.0%	8.54 ± 2.18	2.20 ± 0.50	0.280 ± 0.30	0.087 ± 0.044
MLP policy (2 steps) + dynamic guiding goal	76.8%	7.94 ± 1.98	1.97 ± 0.39	0.278 ± 0.27	0.090 ± 0.044

workspace clear while keeping the robot working efficiency. Here we did not consider moving obstacles other than humans in the environment, such as the cloth, which can be further considered in the future work.

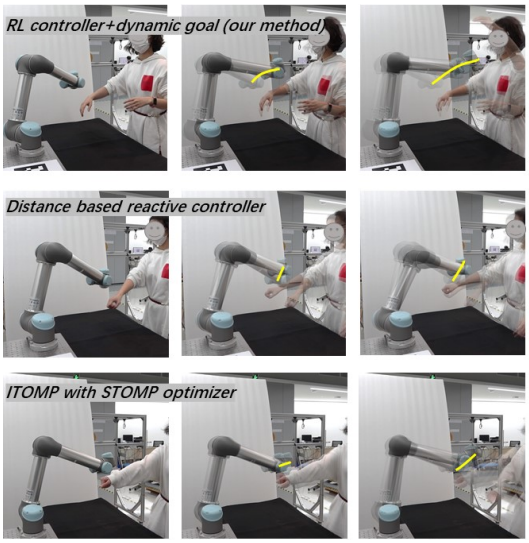

Fig. 10: The snapshot of the comparison result among three methods with same reference path in a fixed time range. Upper: our method where the robot successfully bypasses human. Middle: the reactive controller where the robot moves away from the human, but its workflow is completely interrupted. Lower: the replanning method where the robot spends longer solving time.

VI. Conclusion and Limitation

In this work, we have suggested a human-aware robot motion adaption controller that can responsively adapt robot motion to balance between task following and human safety. By using our RL based controller with a hazard evaluation based dynamic goal selection, the robot can fast avoid human without interrupting its own workflow, has a higher task success rate, and better balances between keeping human safe and following demonstrated path. Since most existing collaborative robots have the function to follow a demonstration path, they can conveniently benefit from our method by combining traditional control with our new method to make a good trade-off between following demonstrations and adapting to human movements. The current real-world test is a usability test and that further work will establish the system performance.

Our work has several limitations that deserve further exploration in the future. First, the user preference is important in the HRC system, e.g., one participant in our real-world experiment reported that even though the RL based method is efficient but it seems to be too aggressive for him. Such user preference can be considered by training multiple policies with different levels of aggressive behaviours and then switching among policies according to the user feedback. Second, we assume that the human state measurement is perfect, but since the human is often occluded by the robot, the measurement can be very noisy. We plan to evaluate the measurement uncertainty and consider it in the policy training. Third, our method still may fail since the reward only poses a soft constraint. But it is difficult to understand why it fails since the interpretable DRL is still an open problem. Finally, the robot output velocity shall be lower if the goal is closer to the robot's current position, especially when the distance is less than $3\,\mathrm{cm}$. A possible solution is to combine traditional control with our method for a more precise control.

References

[1] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 1478–1483.

[2] C. Park, J. Pan, and D. Manocha, "ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments," in *International Conference on Automated Planning and Scheduling*, 2012.

[3] J. S. Park, C. Park, and D. Manocha, "I-planner: Intention-aware motion planning using learning-based human motion prediction," *The International Journal of Robotics Research*, vol. 38, no. 1, pp. 23–39, 2019.

[4] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*. Springer, 1986, pp. 396–404.

[5] S. Calinon, I. Sardellitti, and D. G. Caldwell, "Learning-based control strategy for safe human-robot interaction exploiting task and robot redundancies," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 249–254.

[6] F. Flacco, T. Kröger, A. De Luca, and O. Khatib, "A depth space approach to human-robot collision avoidance," in *IEEE International Conference on Robotics and Automation*, 2012, pp. 338–345.

[7] K. Ota, D. K. Jha, T. Oiki, M. Miura, T. Nammoto, D. Nikovski, and T. Mariyama, "Trajectory optimization for unknown constrained systems using reinforcement learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019, pp. 3487–3494.

[8] T. Fan, P. Long, W. Liu, J. Pan, R. Yang, and D. Manocha, "Learning resilient behaviors for navigation under uncertainty," in *IEEE International Conference on Robotics and Automation*, 2020, pp. 5299–5305.

[9] X. Zhao, T. Fan, D. Wang, Z. Hu, T. Han, and J. Pan, "An actor-critic approach for legible robot motion planner," in *IEEE International Conference on Robotics and Automation*, 2020, pp. 5949–5955.

[10] M. El-Shamouty, X. Wu, S. Yang, M. Albus, and M. F. Huber, "Towards safe human-robot collaboration using deep reinforcement learning," in *IEEE International Conference on Robotics and Automation*, 2020, pp. 4899–4905.

[11] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv:1509.02971*, 2015.

[12] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba, "Hindsight experience replay," *Advances in neural information processing systems*, vol. 30, pp. 5048–5058, 2017.

[13] Y. Kim and B. Mutlu, "How social distance shapes human–robot interaction," *International Journal of Human-Computer Studies*, vol. 72, no. 12, pp. 783–795, 2014.

[14] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2019.

[15] D. Kulić and E. Croft, "Pre-collision safety strategies for human-robot interaction," *Autonomous Robots*, vol. 22, no. 2, pp. 149–164, 2007.

[16] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 4569–4574.